Evoluční návrh hardware

Lukáš SEKANINA[1]

Abstrakt. Kapitola shrnuje problematiku evolučního návrhu hardware, kdy je cílem pomocí evolučního algoritmu navrhnout nové výhodné implementace číslicových obvodů, analogových obvodů, antén a dalších zařízení. Na případových studiích zejména z oblasti návrhu číslicových obvodů jsou demonstrovány výhody a limity evolučního návrhu. Pozornost je rovněž věnována teoretické analýze evolučně navržených výpočetních systémů a důsledkům pro tzv. problém implementace. Dále je nastíněno, jakým způsobem by bylo vhodné interpretovat patentové právo, aby bylo užitečné i z pohledu automatického generování patentovatelných vynálezů.

1 Úvod

Použití různých optimalizačních algoritmů se stalou nedílnou součástí metod automatizovaného návrhu a implementace elektronických obvodů. Jednou z nejpopulárnějších metod je *evoluční optimalizace* [3, 13, 35]. V poslední dekádě můžeme pozorovat velký zájem o *genetické programování*, což je metoda spadající do třídy evolučních algoritmů (EA), která dovoluje elektronické obvody nejen optimalizovat, ale i automaticky sestavovat z předem zvolených komponent. Oproti optimalizačnímu přístupu, kdy jsou hledány nejvýhodnější hodnoty předem zvolených parametrů již hotového návrhu, umožňuje genetické programování vytvářet zcela nové struktury. V mnoha případech můžeme přímo hovořit o automatizovaném generování patentovatelných invencí. Počítač využívající evoluční algoritmy začíná významně konkurovat kvalifikovanému a kreativnímu inženýru-návrháři, který by dokonce mohl časem i ztratit zaměstnání [24]. Mezi nejzajímavější výsledky evolučního návrhu z oblasti elektroniky a výpočetní techniky řadíme zejména různé analogové filtry, regulátory, číslicové obrazové operátory, antény, optické systémy a komunikační protokoly (viz shrnující publikace [7, 12, 24]). U konkrétních evolučně navržených realizací bylo prokázáno, že fungují lépe (podle stanoveného kritéria) než nejlepší doposud známá řešení vytvořená konvenčními návrhovými technikami.

Kandidátní řešení produkovaná evolučním algoritmem jsou nejčastěji ohodnocována pomocí simulátoru, který je běžně používán v dané oblasti. Zajímavější

[1] Fakulta informačních technologií, Vysoké učení technické v Brně, Božetěchova 2, 612 66, E-mail: sekanina@fit.vutbr.cz

situace nastává, pokud jsou kandidátní řešení ohodnocována bez použití simulátoru, přímo v reálném prostředí. V oblasti *vyvíjejících se obvodů* (*evolvable hardware*) [7, 14, 22] je možné použít tzv. *rekonfigurovatelné obvody*. Kandidátní jedinec, se kterým pracuje EA, potom představuje buď přímo konfiguraci rekonfigurovatelného obvodu nebo předpis, jak vytvořit kandidátní konfiguraci. Ve fitness funkci je následně testováno, do jaké míry plní kandidátní obvod požadavky zadané uživatelem. Např. se spočítá, pro kolik vstupních kombinací správně pracuje kandidátní kombinační číslicový obvod.

Použití rekonfigurovatelného obvodu namísto simulátoru má několik výhod: (1) Evoluční proces je obvykle výrazně rychlejší než při simulaci kandidátních obvodů v počítači. (2) Pokud je EA implementován přímo v cílové aplikaci, může zajišťovat dynamickou adaptaci na měnící se okolní podmínky, popř. znovuobnovit činnost zařízení, pokud dojde k poškození části čipu. (3) EA může objevit nové implementace zadaného problému, které mohou být principielně mimo prostor řešení, které je expert v daném oboru vůbec schopen vymyslet. Protože evoluční design probíhá v reálném fyzickém prostředí (v konkrétním čipu, za určité teploty, při existenci určitého šumu, elektromagnetického záření, při jistém kolísání napájecího napětí apod.), může dojít k nalezení řešení, které je perfektně adaptováno pro daný čip, prostor a čas. Adrian Thompson jako první ukázal, že je EA schopen najít velmi neobvyklá (a nepochopitelná) řešení, která jsou mimo dosah konvenčních návrhových technik a která *nefungují*, pokud jsou přemístěna z prostředí, kde vznikla (např. do jiného rekonfigurovatelného obvodu) [28].

Protože můžeme evolučně konstruovat elektronické obvody, můžeme rovněž konstruovat výpočetní systémy, přesněji řečeno, fyzická výpočetní zařízení. V této kapitole rovněž ukážeme, že výpočetní systémy navržené a fyzicky realizované evolučními technikami vykazují vlastnosti, jenž nenajdeme v existujících výpočetních systémech, které jsou běžně navrhovány inženýry. Evolucí navržené systémy provádějí požadované výpočty, ale často není možné rozpoznat, jak a na jakém principu pracuje fyzická implementace. Důvodem je fakt, že evoluce využívá pro konstrukci těchto zařízení fyzikální a chemické vlastnosti materiálů, ze kterých jsou zařízení realizována, a různé okamžité charakteristiky prostředí, jako jsou např. teplota, elektromagnetické pole atd. Podobně je tomu např. u mozku, který je adaptován na určité prostředí a pouze v tomto prostředí funguje. Chování mozku lze také interpretovat jako výpočetní proces, ale doposud přesně nevíme, jak je tento „výpočet“ prováděn. Problémem tedy je, že neexistuje abstraktní model výpočtu a zobrazení mezi abstraktním výpočetním procesem a jeho fyzickou implementací. Můžeme se ptát, zda lze vůbec považovat evolucí vytvořená výpočetní zařízení za *výpočetní mechanismy* (computing mechanisms [19]) ve smyslu Turingova stroje.

Cílem této kapitoly je podat základní přehled o evolučním návrhu obvodů, jak s využitím simulátoru obvodů, tak i pomocí rekonfigurovatelných obvodů. Bude nás zajímat, do jaké míry může evoluční návrh konkurovat konvenčnímu návrhu elektronických obvodů (zejména v oblasti číslicových obvodů). Další část kapitoly je věnována teoretické analýze problému implementace výpočetního zařízení z pohledu evolučního návrhu. Budeme charakterizovat třídu evolučně navržených výpočetních zařízení z pohledu existujících výpočetních zařízení navržených konvenční cestou a z

pohledu živých systémů, které existují v přírodě a jimž často rovněž přisuzujeme výpočetní schopnosti. V poslední části kapitoly se zaměříme na problematiku patentování evolučně vytvořených řešení.

2 Evoluční návrh

Evoluční návrh vychází z metod evoluční optimalizace. V této kapitole vysvětlíme princip metody a popíšeme použití kartézského genetického programování pro návrh číslicových obvodů.

2.1 Problém návrhu jako problém prohledávání

Evoluční algoritmy jsou stochastické prohledávací algoritmy inspirované Darwinovou teorií evoluce a neodarwinismem [2, 13, 35]. Kandidátní řešení je reprezentováno řetězcem symbolů, který v EA nazýváme chromozom nebo genotyp. V případě evoluční optimalizace se jedná o sadu optimalizovaných parametrů, které jsou vhodně zakódovány. V případě evolučního návrhu je řešení kódováno buď *přímo*, např. jako seznam použitých komponent, s informací o jejich propojení a parametrech, nebo *nepřímo*, např. jako program, jehož vykonáním dojde k sestavení kandidátního obvodu. Množina všech chromozomů tvoří prohledávaný prostor, ve kterém evoluční algoritmus pracuje. Kvalitu kandidátního jedince určuje fitness funkce, která ohodnotí kandidátní fenonotyp vytvořený z příslušného genotypu. Ve fitness funkci je reflektována specifikace problému, který je pomocí EA řešen. Nová kandidátní řešení jsou vytvářena pomocí geneticky inspirovaných operací (mutace, křížení apod.), které pracují nad chromozomy. Selekční tlak vede prohledávací algoritmus k výhodnějším částem prostoru možných řešení.

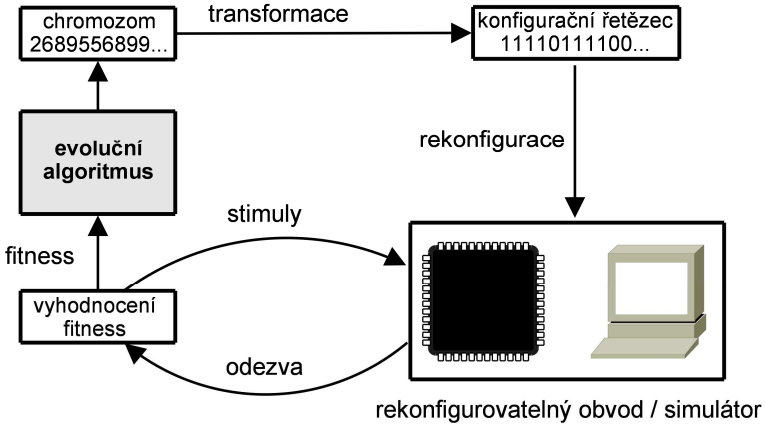
EA pracuje následovně: nejdříve je vygenerována počáteční populace (buď náhodně nebo pomocí heuristiky), jejíž prvky jsou ohodnoceny fitness funkcí. Dále, dokud není splněna ukončující podmínka, jsou prováděny následující kroky:

- Ze staré populace je vytvořena pomocí genetických operátorů nová populace.
- Každý jedinec nové populace je ohodnocen fitness funkcí.

Je tedy zřejmé, že oproti konvenčnímu návrhu je evoluční návrh založen na procesu „vygeneruj řešení a otestuj ho“. Důležité je, že změna probíhající v kandidátním řešení je náhodná. Naopak, konvenční přístup modifikuje existující řešení s určitou představou o výsledku, obvykle podle dobře definovaného a zavedeného postupu a tedy s určitým cílem.

Obrázek 1 ukazuje princip evolučního návrhu elektronického obvodu s využitím rekonfigurovatelného zařízení. Pokud EA pracuje přímo na úrovni konfiguračního řetězce rekonfigurovatelného obvodu, potom odpadá fáze „transformace“, tj. vytvoření konfigurace obvodu podle chromozomu. Evaluace kandidátního řešení představuje proces, který většinou nejvíce ovlivňuje dobu evoluce. Je tedy snaha ji redukovat. Např. evaluace kandidátního analogového obvodu provedená v rekonfigurovatelném

obvodu může být o několik řádů rychlejší než simulátor [25]. Zde se obvykle používají simulátory z rodiny Spice, které simulují přesně, ale pomalu.



Obr. 1. Evoluční návrh s využitím rekonfigurovatelného obvodu. Namísto rekonfigurovatelného obvodu je možné použít simulátor.

Pokud navrhujeme číslicové obvody na úrovni hradel, můžeme v některých případech simulaci obvodu výrazně urychlit „předpočítáním“ určitých hodnot, popř. paralelní simulací. Zde však platí, že s každým přidaným vstupem se doba evaluace zdvojnásobí, pokud ohodnocujeme všechny možné kombinace na vstupech. Proto je tento přístup použitelný pouze pro případy, kdy je počet vstupů malý. Pro složitější obvody musíme definovat *trénovací množinu* vstupních vektorů, která se použije pro evaluaci, a na konci evoluce ověřit výsledné řešení pomocí *testovací množiny*. Typickou aplikací je zpracování signálů a obrazů [22, 24]. Vzhledem k tomu, že není garantováno, že navržený obvod bude pracovat správně pro všechny možné kombinace na vstupech, je třída aplikací této techniky omezena. Ve speciálních případech můžeme použít i jiné principy evaluace kandidátních řešení (kapitola 3.4).

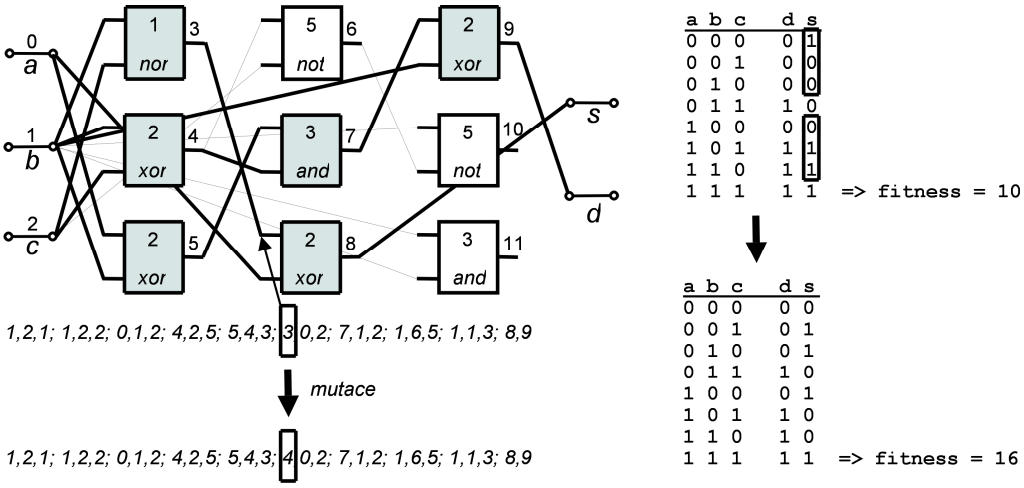
Dalšími faktory, které ovlivňují dobu evaluace kandidátního obvodu, jsou doba rekonfigurace obvodu a doba nutná pro výpočet fitness hodnoty z naměřených dat. Doba nutná pro vytvoření nové populace pomocí genetických operátorů je při dobrém návrhu aplikace většinou zanedbatelná.

2.2 Kartézské genetické programování

Předpokládejme, že je cílem navrhnout schéma zapojení obvodu pro zadanou specifikaci. Následnou realizaci na čipu zatím ponechme stranou. Kartézské genetické programování (Cartesian Genetic Programming – CGP) je vhodné pro evoluční návrh grafových struktur, a tedy i schémat číslicových obvodů [16]. Jedná se o variantu genetického programování, u které jsou kandidátní řešení reprezentována pomocí orientovaných grafů. Kandidátní obvod je modelován jako pole programovatelných elementů (uzlů grafu) o velikosti n_c x n_r (počet sloupců x počet řádků). Označme počet primárních vstupů obvodu n_i a počet primárních výstupů obvodu n_o. Každý z uzlů, jenž

může mít až n_t vstupů, realizuje právě jednu funkci, která je vybrána z množiny dostupných funkcí F. Vstupy uzlu, který se nachází v j-tém sloupci, mohou být připojeny buď na primární vstupy obvodu nebo na výstupy uzlů umístěných v 1 až L předchozích sloupcích. Uzly stejného sloupce se nesmějí propojovat. Primární výstupy mohou být připojeny k libovolnému uzlu obvodu.

Obrázek 2 ukazuje příklad instance CGP. Každý kandidátní obvod je zakódován pomocí řetězce celočíselných hodnot (genotypu). Způsob kódování je následující: Každému primárnímu vstupu obvodu je přiřazen index z intervalu 0, …, n_i -1. K výstupům uzlů jsou postupně rovněž přiřazeny indexy, a to po sloupcích, s počáteční hodnotou n_i pro nejlevější horní uzel. Každý uzel obvodu je kódován pomocí n_t+1 celočíselných hodnot. Prvních n_t hodnot určuje indexy uzlů, ke kterým budou připojeny vstupy uvažovaného uzlu. Poslední hodnota určuje kód logické funkce uzlu. Na konci chromozomu najdeme n_o hodnot definujících indexy uzlů, ke kterým jsou připojeny primární výstupy obvodu. Základní vlastností uvedeného kódování je, že zatímco délka chromozomu je vždy konstantní, velikost zakódovaného obvodu (fenotypu) je variabilní. Množina všech chromozomů určuje prohledávací prostor, ve kterém pracuje evoluční algoritmus.



Obr. 2. Příklad instance CGP s parametry n_c = 3, n_r = 3, n_i = 3, n_o = 2, n_t = 2, odpovídajícího genotypu a pravdivostní tabulky. Pokud je cílem navrhnout úplnou sčítačku (s operandy *a* a *b*, *c* je vstupní přenos, na výstupu *s* označuje součet a *d* je výstupní přenos), je získána fitness hodnota 10. Po provední jediné mutace (3→4 na pozici 16) je možné získat plně funkčí sčítačku s fitness hodnotou 16.

Základní varianta prohledávacího algoritmu, která je využita v CGP, používá populaci o velikosti 1+λ jedinců (obvykle λ = 4) a jediný genetický operátor – mutaci, který pracuje tak, že náhodně vybere h genů (tj. h hodnot z chromozomu) a náhodně vygeneruje jejich nové hodnoty (obrázek 2 ukazuje příklad pro h = 1). Pomocí mutace vznikají nová kandidátní řešení problému. Počáteční populace je vygenerována náhodně nebo ji tvoří předem připravená řešení problému. Kvalita každého

kandidátního řešení je ohodnocena pomocí fitness funkce. V případě návrhu malých kombinačních obvodů je například vypočteno, kolik je schopen kandidátní obvod vyprodukovat správných bitů pro všechny možné kombinace vstupních signálů. Pro obvod na obrázku 2, který má tři vstupy a dva výstupy, je fitness hodnota 10. Po provedení jediné mutace je však možné získat plně funkční úplnou sčítačku s fitness hodnotou 16. Nová populace je vytvářena tak, že je ze staré populace vybrán jedinec s nejlepší hodnotou fitness, který je vložen do nové populace spolu se svými λ kopiemi pozměněnými pomocí mutace. V CGP se používá jedno důležité pravidlo: Pokud existuje více "nejlepších" jedinců se stejnou fitness hodnotou, použije se jako rodič ten, který nebyl rodičem v předchozí generaci. Tímto je podporována genetická diverzita populace. Evoluce končí nalezením dostatečně kvalitního jedince nebo vyčerpáním povoleného počtu generací.

Kartézské genetické programování bylo použito nejen pro návrh kombinačních obvodů, ale i v úlohách symbolické regrese, filtrace a klasifikace, pro realizaci kontrolérů, developmentálních zobrazení, sebemodifikujícího se kódu atd. [24].

3 Příklady evolučního návrhu

Abychom demonstrovali různé přístupy k reprezentaci problému a výpočtu fitness hodnoty, ukážeme, jak je možné pomocí evolučního algoritmu navrhovat malé kombinační obvody (budou obsahovat desítky hradel), obrazové filtry (tisíce hradel) a testovací obvody (miliony hradel).

3.1 Soutěž Humies

Evolučně navržené obrazové filtry i testovací obvody byly prezentovány v rámci soutěže Humies (Human competitive awards in genetic and evolutionary computation [8]), která je pořádána každoročně na konferenci GECCO – nejvýznamnější konferenci o evolučních a genetických algoritmech. V rámci soutěže je třeba představit výsledek, který byl získán pomocí technik evolučního návrhu – tedy automaticky vygenerován počítačem simulujícím evoluční proces. Komise složená z předních odborníků posuzuje, zda je takto vytvořený výsledek schopen soutěžit s lidskou invencí. Evolučně navržené obrazové filtry získaly v roce 2004 čestné uznání a testovací obvody byly v roce 2008 oceněny stříbrnou medailí. Na stránkách Humies [8] najdeme desítky výsledků, které byly vyprodukovány evolučními algoritmy a které jsou konkurenceschopné výstupům inženýra-návrháře. Kdy je tedy výsledek shledán jako „human-competitive“? Musí splňovat alespoň jedno z uvedených kritérií, které zavedl John Koza [12]:

- Výsledek byl patentován jako vynález v minulosti, je vylepšením patentovaného vynálezu nebo by dnes mohl být kvalifikován jako patentovatelný vynález.
- Výsledek je stejně dobrý nebo lepší než jiné řešení, které bylo akceptováno jako nový vědecký výsledek v době, kdy bylo publikováno v recenzovaném vědeckém časopisu.

- Výsledek je stejně dobrý nebo lepší než jiný výsledek, který byl vložen do databáze nebo archivu výsledků udržovaných mezinárodně uznávanou skupinou vědců.
- Výsledek je publikovatelný ve své podstatě jako nový vědecký výsledek nezávisle na tom, že byl mechanicky vytvořen.
- Výsledek je stejně dobrý nebo lepší než nejaktuálnější člověkem vytvořené řešení dlouho známého problému, pro který vznikala řada neustále lepších výsledků navržených člověkem.
- Výsledek je stejně dobrý nebo lepší než to řešení, které bylo považováno za pokrokové ve svém oboru, když bylo objeveno.
- Výsledek řeší problém, který má pro daný obor nepopiratelnou obtížnost.
- Výsledek zvítězil v soutěži s řešeními vytvořenými člověkem (buď ve hře proti člověku, nebo proti programu, který byl napsán člověkem).

3.2 Kombinační obvody

Předpokládejme, že máme obvodově realizovat logickou funkci o n vstupech, která je definována pomocí pravdivostní tabulky. Z pravdivostní tabulky je možné odvodit kanonický tvar v uvažovaném výpočetním modelu (např. výraz v normální disjunktní formě, úplný binární rozhodovací diagram apod.). Cílem logické syntézy je obvykle najít takové vyjádření logické funkce v uvažovaném modelu, které minimalizuje cenu realizace, tj. počet hradel, zpoždění apod. V rámci zvoleného modelu existuje množina transformací, které je možné na logický výraz aplikovat a tím měnit jeho tvar. Nikdy však nemůže dojít ke změně evaluace logické funkce, kterou výrazy reprezentují. Zvolený model a s ním spojené transformace určují prostor možných řešení. Konvenční algoritmy (např. Espresso, ABC apod.), které se v rámci syntézy používají, jsou navrženy tak, že dokáží nalézt co nejvýhodnější řešení, ale pouze v rámci zvoleného modelu, tj. v předem vymezeném podprostoru možných řešení.

Evoluční návrh vychází z toho, jaké zdroje jsou dostupné pro implementaci na konkrétní platformě. Jedná se zejména o typ výpočetních elementů, jejich možné funkce, počty vstupů a výstupů a možnosti propojování. Podle konkrétní cílové platformy je potom vytvořena taková reprezentace obvodu, která by měla umožnit postihnout libovolný obvod z množiny všech vytvořitelných obvodů. Tím je definován prohledávací prostor pro evoluční algoritmus. V rámci tohoto prostoru mohou existovat velmi výhodné implementace požadované logické funkce, které nejsou dosažitelné v konvenčních modelech.

Jednou z prvních aplikací CGP byl evoluční návrh kombinačních sčítaček a násobiček, které obsahují co nejmenší počet hradel. Pro tuto úlohu je fitness f definována

$$f = B + (n_r.n_c - g), \tag{1}$$

kde B je počet správně určených výstupů (měřeno pro všechny vstupní kombinace, tj. maximální hodnota $B_m = n_o 2^{ni}$ pro $n = n_i$) a g je počet použitých hradel. Zde je nutné

poznamenat, že počet se výraz hradel ($n_r.n_c - g$) začíná započítávat až v okamžiku, kdy B dosáhne maximální hodnoty B_m.

Tabulka 1 shrnuje nastavení experimentů a získané výsledky podle [16, 30]. Je zřejmé, že CGP dokáže najít násobičky (pro dvou až čtyřbitové operandy), které obsahují v průměru méně dvouvstupových hradel než násobičky získané konvenčními algoritmy.

Tabulka 1. Počet hradel v násobičkách navržených konvenčně a pomocí CGP (nejlepší ze 100 nezávislých běhů). CGP pracuje s velikostí populace 5 jedinců, h = 3, L= n_c, množina hradel obsahuje funkce x AND y, x XOR y a (not x) AND y.

Násobička	Nejlepší konvenční	Nejlepší CGP	n_r x n_c	Max. generací
2b x 2b	8	7	1 x 7	10 tis.
3b x 2b	17	13	1 x 17	200 tis.
3b x 3b	30	23	1 x 35	20 mil.
4b x 3b	47	37	1 x 56	200 mil.
4b x 4b	64	57	1 x 67	700 mil.

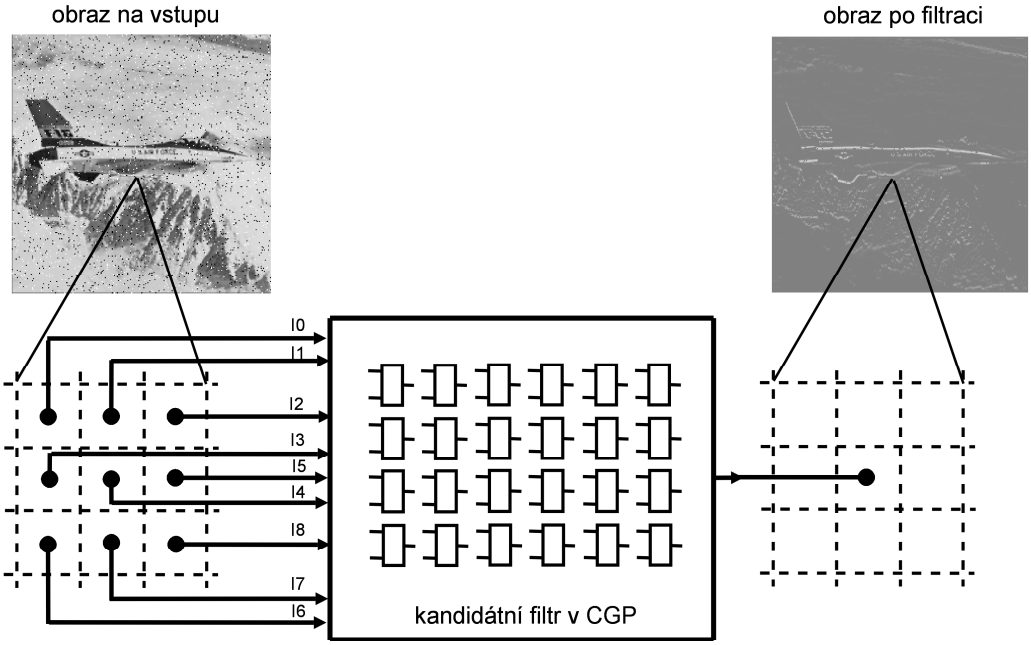
Za hlavní nevýhodu evolučního návrhu je považován fakt, že výše popsaná metoda funguje jen pro relativně jednoduché obvody. Pokud totiž budeme zvyšovat počet vstupů obvodu a současně budeme požadovat ověření činnosti obvodu pro každý možný vstupní vektor, poroste doba evaluace kandidátního řešení exponenciálně. Dalším problémem je prodlužování chromozomu, které je způsobeno požadavkem na vytváření velkých obvodů. Dlouhé chromozomy implikují velké prohledávací prostory, ve kterých evoluční algoritmy nedokáží efektivně nacházet zajímavá řešení. Pro obvody s cca 10-15 vstupy již evoluce nenajde rozumné řešení vůbec. Proto byly zavedeny různé techniky, které umožňují se výše uvedeným problémům vyhnout, např. inkrementální evoluce, modulární přístup apod. [24].

Z praktického pohledu je užitečné nezačít evoluční proces z náhodně vygenerované populace, ale z již existujícího řešení získaného konvenční metodou (např. ABC, SIS apod.). Pomocí CGP je potom možné v konvenčním řešení redukovat počet hradel i o desítky procent.

3.3 Evoluční návrh obrazových operátorů

K potlačení šumu v obraze nebo k detekci hran se nejčastěji používají nelineární číslicové filtry, které pracují tak, že novou hodnotu filtrovaného pixelu počítají pomocí vhodné nelineární funkce na základě předešlé (potenciálně poškozené) hodnoty pixelu a jeho nejbližších sousedů. Obrázek 3 ukazuje příklad filtru, jenž používá 9 pixelů na vstupu, které spolu tvoří tzv. filtrační masku o velikosti 3x3 pixely. Pokud budou pixely reprezentovány na 8 bitech, můžeme takový filtr chápat jako číslicový obvod s 9x8 bitovými vstupy a 8 bitovým výstupem. V praxi se pro potlačení lineárního šumu běžně používají průměrovací (váhové) filtry, pro potlačení výstřelového šumu různé varianty mediánových filtrů a pro detekci hran potom specializované operátory, např.

Sobelův nebo Cannyho detektor [27]. Hlavním cílem pro evoluční návrh v této oblasti je najít co nejlépe filtrující operátor pro konkrétní typ šumu a současně takové řešení, jehož implementace na čipu zabere méně plochy než zmíněné konvenční implementace.



Obr. 3. Použití CGP pro návrh obrazových filtrů. „Obraz po filtraci“ je příkladem výstupu jednoho z evolučně navržených filtrů, který dokáže současně potlačit výstřelový šum a detekovat hrany.

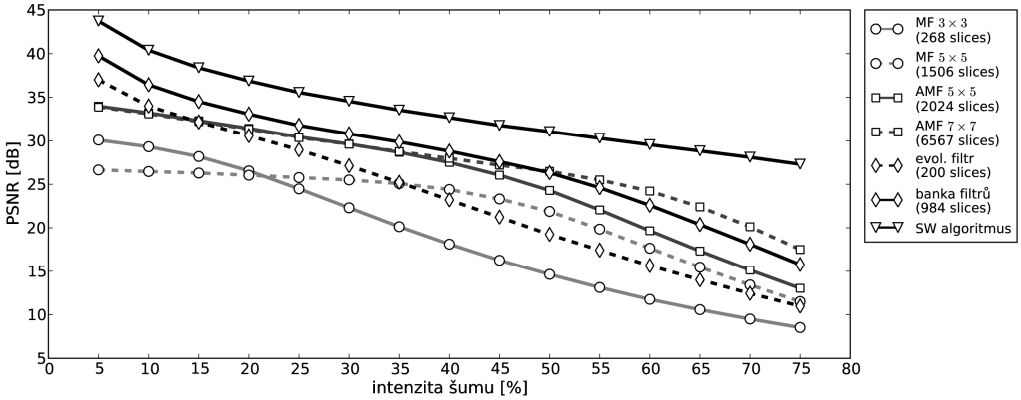
Evoluční návrh nelineárního filtru, který má potlačit např. výstřelový šum a současně detekovat hrany v obraze (viz výstup filtru na obrázku 3), může být proveden pomocí CGP. Kandidátní filtr však nebudeme reprezentovat na úrovni hradel, ale na úrovni funkčních bloků, které mají dva 8-bitové vstupy a jeden 8-bitový výstup a které realizují vhodné funkce, např. průměr, minimum, maximu apod. (konkrétní množina funkcí je uvedena např. v [22, 31]). Reprezentace na úrovni funkčních bloků umožňuje pracovat s relativně krátkým chromozomem (tj. malým prohledávacím prostorem), který by při práci na úrovni hradel několikanásobně narostl. Typicky je použito pole velikosti 8 x 4 funkčních bloků, populace obsahuje 8 jedinců, $h = 1$, $L = 1$ a je vytvářeno 10-100 tisíc generací [22, 24].

Při evaluaci kandidátního filtru není možné ověřit odezvu filtru pro všechny možné vstupní kombinace pixelů. Proto je v průběhu evoluce použit vhodně zvolený trénovací obraz. Cílem evoluce je najít filtr, který dosahuje co nejmenší odchylku ideální (nepoškozené) verze obrázku v od obrázku, který je filtrem generován. Pokud označíme obrázek generovaný tímto kandidátním filtrem jako w, potom je fitness hodnota kandidátního filtru definována jako

$$fitness = \sum_{i=1}^{M} \sum_{j=1}^{N} | v(i, j) - w(i, j) |, \quad (2)$$

kde N a M jsou rozměry obrázku, typicky $M = N = 128$. Funkčnost a obecnost evolučně navrženého filtru je na konci evoluce ověřena pomocí sady testovacích obrázků.

V publikacích [22, 24, 31] bylo prokázáno, že evolučně navržené filtry pro různé typy šumu (např. sůl a pepř, dávkový výstřelový šum) a detektory hran dosahují minimálně stejné vizuální kvality výsledku ve srovnání s konvenčním řešením (mediánové filtry, Sobelův detektor apod.). Navíc tyto filtry, pokud by byly implementovány na čipu, zaberou podstatně méně plochy než konvenční řešení. Na obrázku 4 je znázorněna průměrná „vizuální kvalita“ vyfiltrovaného obrazu pomocí několika různých přístupů v závislosti na intenzitě impulsního šumu typu sůl a pepř. Průměrná vizuální kvalita byla získána na sadě 25 testovacích obrazů [24]. Evolučně navržený filtr (*evol. filtr*) je sice mnohem kvalitnější než konvenční mediánový filtr (*MF*) s filtrovacím jádrem 3x3, ale kvalita filtrace značně klesá se vzrůstající intenzitou šumu. Použijeme-li mediánový filtr s větším filtrovacím oknem, je evolučně navržený filtr rovněž kvalitnější pouze do určité intenzity šumu. Navíc existuje propracovanější konvenční metoda – adaptivní mediánový filtr (*AMF*) – který poskytuje mnohem kvalitnější výsledky než klasický mediánový filtr [9]. Adaptivní medián pracuje s větším filtrovacím jádrem než 3x3 pixely a tudíž je i jeho obvodová realizace nákladnější (viz cena na obrázku 4). Pro vyšší intenzitu šumu se pomocí evolučního postupu kvalitnější filtr navrhnout nedaří, což je dáno tím, že je použito filtrovací jádro obsahující pouze 3x3 obrazové body. Nejlepší známá metoda pro odstranění šumu typu sůl a pepř [10], jejíž výsledek byl rovněž začleněn do obrázku 4 (*SW algoritmus*), není vhodná pro porovnání, protože neprovádí filtraci na základě lokální informace, ale na základě celého obrázku. Tento přístup je nevhodný pro obvodovou akceleraci.



Obr. 4. Průměrná kvalita filtrace vypočtená na sadě 25 obrázků, která byla dosažena různými filtry pro různou intenzitu šumu typu sůl a pepř. Cena filtru je vyjádřena počtem slices – konfigurovatelných elementů FPGA (viz kapitola 4.1).

Evoluční návrh však má jednu unikátní vlastnost: schopnost generovat odlišná a přitom obdobně kvalitní řešení. Nalezneme-li více (např. 3 nebo 5) kvalitních filtrů s

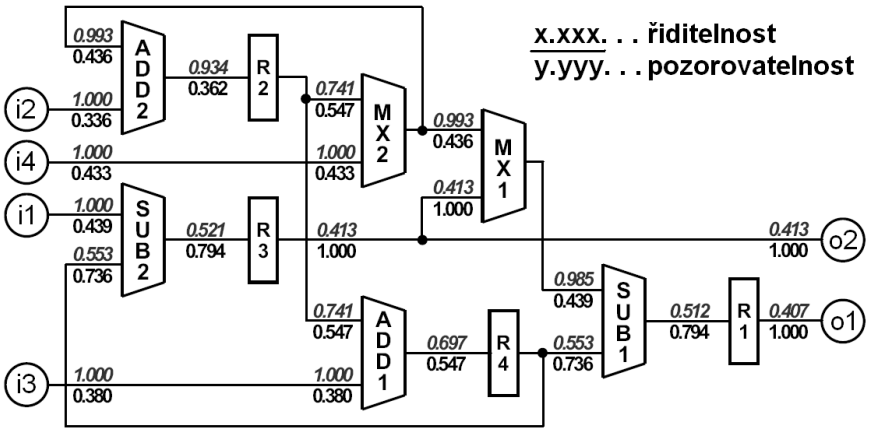
jádrem 3x3 a necháme-li je filtrovat obraz paralelně, je velmi pravděpodobné, že většina z nich bude pro všechny devítice pixelů dávat na výstupu správnou hodnotu. Vytvoříme-li tedy obvod, který z výsledků těchto filtrů zvolí nejlepší řešení (např. pomocí mediánu), můžeme realizovat filtr (na obrázku 4 označený jako *banka filtrů*), který je schopen odstranit i šum s vyšší intenzitou. Filtr tímto způsobem vytvořený zabírá mnohem menší plochu než adaptivní mediánový filtr, přičemž dosahuje srovnatelné vizuální kvality výsledku. Evolučně navržená banka filtrů je chráněna v České republice užitným vzorem UV20017/2009 (patentová přihláška byla rovněž podána).

3.4 Evoluční návrh testovacích obvodů

Kromě bezvadné funkčnosti a nízkého příkonu je jedním z klíčových požadavků na současné složité číslicové obvody i snadná *testovatelnost* [17]. Obvody většinou podporují diagnostický režim, po jehož aktivaci je možné detekovat poruchy jak interních prvků tak i spojů. V diagnostickém režimu jsou na primární vstupy obvodu postupně připojovány vhodně zvolené hodnoty (tzv. testovací posloupnost, zkráceně *test*) a je sledována odezva na primárních výstupech, která je následně porovnána s předem vypočtenými referenčními hodnotami. Ideální test by měl obsahovat co nejmenší množinu testovacích vektorů a detekovat všechny požadované poruchy.

Pro návrháře číslicových obvodů je velmi důležité zjistit, jak obtížné bude vytvořit *test* pro navržený obvod, popř. přímo identifikovat obtížně testovatelné části obvodu. Pro tyto účely se používají *metody analýzy testovatelnosti* obvodů. Jejich výhodou je nižší časová složitost oproti metodám přímo generujícím test. Obecně přijímaná definice testovatelnosti bohužel neexistuje. Existuje však řada dílčích, konkrétně aplikačně orientovaných definic a z nich vycházejících metod, z nichž každá je obvykle konstruována pro jistou konkrétní úroveň popisu obvodu a pro zohlednění vybrané podmnožiny nákladů spojených s jeho testováním. Aby bylo možné existující a nově navrhované metody analýzy testovatelnosti posuzovat, je nutné disponovat sadou různě složitých testovacích obvodů, u kterých je známa míra testovatelnosti. Sada testovacích obvodů s požadovanými vlastnostmi a složitostí v této oblasti doposud neexistovala.

Práce [18] ukazuje, že je evoluční přístup schopen vytvářet obvody (sestávající po syntéze z milionu i více hradel), které mají požadovanou míru testovatelnosti a jsou tak reálně použitelné pro posuzování kvality metod analýzy testovatelnosti. Přístup použitý pro ohodnocení testovatelnosti kandidátního obvodu je založen na ohodnocení parametrů *řiditelnosti* a *pozorovatelnosti*. Řiditelnost resp. pozorovatelnost je chápána jako schopnost ovlivnit resp. změřit hodnotu signálu v daném místě obvodu. Řiditelnost resp. pozorovatelnost je vyjádřena hodnotou v rozsahu <0, 1>. Hodnota řiditelnosti resp. pozorovatelnosti pak obvykle vyjadřuje míru snadnosti nastavení resp. zjištění hodnoty signálu v daném místě obvodu. Pro účely ohodnocení kandidátního obvodu je v našem případě vypočtena průměrná řiditelnost resp. pozorovatelnost všech vstupů resp. výstupů u všech prvků obvodu. Obrázek 5 ukazuje příklad obvodu s vypočtenými hodnotami řiditelnosti a pozorovatelnosti jednotlivých signálů obvodu podle metody [18].



Obr. 5. Číslicový obvod s ohodnocenou řiditelností a pozorovatelností

Vstupem pro evoluční návrh je uživatelem zadaný počet primárních vstupů a výstupů obvodu, počet a typ prvků obvodu, požadavek na průměrnou řiditelnost a pozorovatelnost a parametry algoritmu. Registry jsou vloženy automaticky na vhodná místa. Např. pro obvod na obrázku 5 by bylo zadáno:

- Počet vstupů: 4 x 8 bitů
- Počet výstupů: 2 x 8 bitů
- Komponenty: 2 x 8bit. odčítačka (SUB), 2 x 8bit. sčítačka, 2x 8bit. multiplexor
- Požadovaná průměrná testovatelnost: 0,75
- Požadovaná průměrná řiditelnost: 0,75
- Velikost populace: 30
- Mutace: 0,02%
- Nahrazení populace: 95%
- Generací: 200

Vzhledem k tomu, že je uživatelem definován počet a typ komponent, obsahuje chromozom pouze informaci o propojení těchto komponent a připojení vstupů a výstupů. Algoritmus evolučního návrhu potom generuje obvody sestavené ze zadaných komponent a snaží se minimalizovat rozdíl mezi požadovanou řiditelností resp. pozorovatelností a řiditelností resp. pozorovatelností změřenou u konkrétního kandidátního jedince.

Zde je důležité upozornit na fakt, že fitness funkcí není vůbec posuzována funkčnost obvodu. Výstupem jsou syntetizovatelné obvody s požadovanou úrovní testovatelnosti a s požadovaným počtem hradel. Metoda byla použita k návrhu sady testovacích obvodů s různým počtem hradel a s různými parametry pozorovatelnosti a řiditelnosti, která je k dispozici zájemcům na Internetu (viz [18]). Nejsložitější z navržených obvodů obsahuje po syntéze 1,2 milióny hradel. Tento obvod je rovněž

největším obvodem, který byl (dle vědomí autorů) vytvořen pomocí evolučního návrhu.

3.5 Kdy je vhodné použít evoluční návrh?

U uvedených příkladů je zajímavé, že evoluční algoritmus pracuje s přibližně stejně velkým chromozomem (tj. v přibližně stejně velkém prohledávacím prostoru), který má velikost maximálně několik tisíc bitů. Další zvětšování velikosti chromozomu je neúnosné a evoluční prohledávání přestává poskytovat rozumné výsledky. Fakt, že evoluce může vytvořit užitečný obvod obsahující miliony hradel, je způsoben *znalostí*, která je dodána do metody řešení. Zatímco při evoluci na úrovni hradel návrhář prakticky nic nepředpokládá o výsledném řešení, u evoluce na úrovni funkčních bloků musí dodat relativně mnoho znalosti. Pokud chceme navrhovat složité obvody, je nutné, aby nemusely být v rámci výpočtu fitness hodnoty vypočteny odezvy pro všechny vstupní kombinace. Pokud stačí pracovat pouze s trénovací množinou, je možné zkrátit dobu výpočtu. Úspěch evolučního algoritmu můžeme očekávat zejména tam, kde je z různých důvodů neúplně nebo nepřesně zadaná specifikace. Obecně je možné konstatovat, že evoluční návrh je nejúspěšnější v těch oblastech, pro které platí [24]:

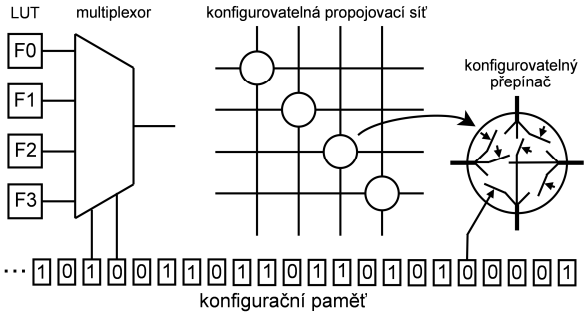
- Konvenční návrhová metoda není plně automatizovatelná a je z části založena na zkušenosti, experimentování a ad hoc postupech.
- Existuje kvalitní simulátor pro ohodnocení kandidátních řešení.
- Evaluace kandidátního řešení není příliš časově náročná.

4 Evoluce přímo v hardware

Tato podkapitola prezentuje některé příklady evolučního návrhu, kdy jsou kandidátní řešení ohodnocována ve fyzickém zařízení, které je umístěno v reálném prostředí.

4.1 Rekonfigurovatelné obvody

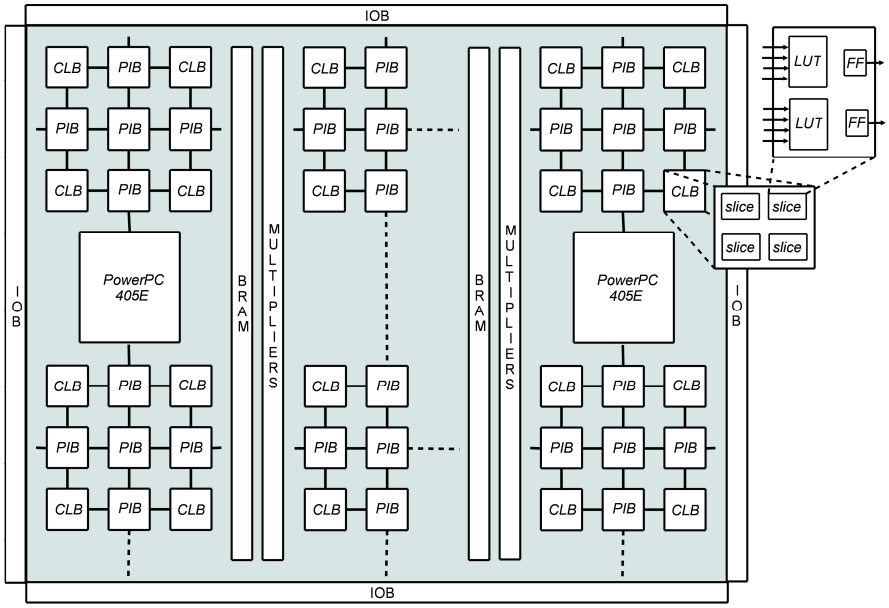
Rekonfigurovatelný obvod obvykle obsahuje pole programovatelných elementů, jejichž funkce, propojení a připojení k primárním vstupům a výstupům je definováno obsahem konfigurační paměti. Tato paměť je nejčastěji typu RAM a její obsah může být relativně rychle modifikován. Obrázek 6 ukazuje, že obsah této paměti ovládá programovatelné propojky, které určují funkci zařízení.



Obr. 6. Princip rekonfigurace propojení číslicového obvodu. Multiplexor připojí dle nastavení konfiguračních bitů jednu z funkčních jednotek F0-F3 na svůj výstup (vlevo). Propojovací síť je realizována pomocí konfigurovatelných přepínačů.

Hradlová pole FPGA

V oblasti číslicových obvodů patří mezi nejpopulárnější rekonfigurovatelné obvody programovatelná hradlová pole FPGA. Typická struktura FPGA firmy Xilinx je uvedena na obrázku 7. Jedná se o dvourozměrné pole programovatelných prvků, které obsahuje konfigurovatelné logické bloky (CLB), programovatelné propojovací bloky (PIB) a konfigurovatelné vstupně/výstupní bloky (IOB). CLB sestává z tzv. *slice*, přičemž každý slice obsahuje generátory logických funkcí realizované pomocí náhledových tabulek (lookup table, LUT) s 3 až 6 vstupy (v závislosti na variantě produktu), klopné obvody a pomocnou logiku. Konfigurační soubor je uložen v paměti SRAM, která je součástí čipu.



Obr. 7. Koncepční schéma obvodu FPGA řady Virtex Pro [33]

Obvody FPGA se liší množstvím a typem konfigurovatelných prvků, které najdeme na čipu. Nejmodernější FPGA obsahují více než 10 tisíc CLB a navíc další specializované komponenty, jako paměti SRAM, rychlé násobičky, procesory (viz procesor PowerPC na obrázku 7), gigabitová rozhraní atd. Integrace těchto specializovaných komponent přímo na čip je důsledkem faktu, že jsou často používány, a bylo by neekonomické, aby je každý uživatel implementoval pomocí CLB a dalších standardních zdrojů.

Obvody FPGA mohou být konfigurovány externě nebo interně. V případě externí rekonfigurace, což je nejběžnější způsob, jak konfigurovat FPGA, jsou konfigurační data nahrána z externí paměti. Interní rekonfigurace je v FPGA řady Virtex podporována pomocí rozhraní ICAP (Internal Access Configuration Port), které umožňuje číst/zapisovat konfigurační data přímo kontrolérem umístěným na čipu. Některé obvody FPGA podporují dynamickou částečnou rekonfiguraci, při které je možné rekonfigurovat část čipu, zatímco jiná část čipu provádí výpočet. Podpora interní rekonfigurace je nutnou podmínkou pro realizaci adaptivního hardware na jednom čipu. Z pohledu evoluční elektroniky je nevýhodné, že výrobci FPGA neposkytují popis konfiguračního souboru FPGA.

Vzhledem k tomu, že je proces návrhu obvodů pro FPGA velmi podobný programování, je možné vytvořit výslednou aplikaci relativně rychle. Návrhář musí nejdříve popsat strukturu obvodu nebo jeho chování pomocí specializovaného jazyka (např. VHDL, Verilog, popř. stačí použít i vhodně rozšířený jazyk C – např. SystemC). Následně je zdrojový kód téměř automaticky přetransformován na konfigurační data pro FPGA. Tento proces, jenž zahrnuje syntézu, rozmístění komponent a jejich propojení v FPGA, je prováděn specializovanými CAD nástroji. Pro syntézu je možné zavést další omezení, např. definovat maximální povolenou velikost obvodu, optimalizovat pracovní frekvenci apod. Rovněž je možné simulovat mezivýsledky transformace a v případě nesplnění požadavků provést změny v návrhu a spustit transformaci znovu.

Rekonfigurovatelné analogové obvody

V oblasti analogových obvodů rovněž existují rekonfigurovatelné čipy. Nejprve uvedeme seznam nejpoužívanějších zkratek:

- FPAA (Field-Programmable Analog Array) – integrovaný obvod, který může být naprogramován tak, aby implementoval analogový obvod. Využívá flexibilní analogové bloky a konfigurovatelné propojení.
- FPMA (Field-Programmable Mixed-Analog-Digital Array) – integrovaný obvod, který obsahuje FPAA, FPGA a konfigurovatelné konvertory za účelem implementace obvodů pracujících se smíšenými signály.
- FPTA (Field-Programmable Transistor Array) – varianta FPAA, kdy základním konfigurovatelným blokem je tranzistor.
- CAB (Configurable Analog Block) – základní programovatelná buňka FPAA.

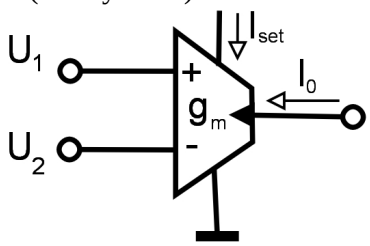
FPAA obsahují pole bloků CAB, programovatelnou propojovací síť, konfigurovatelné I/O a další podpůrné obvody. Na čipu je rovněž umístěna konfigurační paměť. Pro

konfiguraci zapojení jsou použity buď tranzistorové spínače (přenosová hradla), spínané kapacitory nebo transkonduktanční operační zesilovače. Stručně vysvětleme princip změny konfigurace pomocí transkonduktančního operačního zesilovače, protože v současnosti vede na nejzajímavější aplikace.

Transkonduktanční zesilovače (v literatuře někdy označované také zkratkou OTA) jsou součástky podobné operačním zesilovačům. Jejich výstup však není napěťový, ale je proudový. U transkonduktančního zesilovače proto nemůžeme hovořit o napěťovém či proudovém přenosu, ale hovoříme o přenosové vodivosti (transkonduktanci). Vztah mezi výstupním proudem I_o a vstupními napětími na inverujícím (U_-) a neinvertujícím (U_+) vstupu je tedy:

$$I_o = -g_m(U_+ - U_-),$$

kde g_m je přenosová vodivost. Ta může být buď pevná (daná vnitřním uspořádáním zesilovače) nebo nastavitelná zvláštním vstupem. Vstupní i výstupní odpor transkonduktančního zesilovače je ideálně nekonečný. Na obrázku 8 je zesilovač OTA s nastavitelnou transkonduktancí, kterou je možné zvolit pomocí proudu I_{set}. Tato varianta se využívá zejména pro realizaci rekonfigurovatelných analogových filtrů, protože velikost I_{set} je možné jednoduše algoritmicky nastavovat pomocí DA převodníku. Hlavní výhodou rekonfigurovatelných obvodů na bázi OTA oproti jiným možnostem rekonfigurace (např. spínaným kapacitorům) je relativně vysoký dosažitelný pracovní kmitočet (stovky MHz).



Obr. 8. Transkonduktanční operační zesilovače s možností konfigurace g_m pomocí I_{set}

4.2 FPGA jako akcelerátor pro evoluční návrh

Obvody FPGA jsou využitelné pro akceleraci evolučního návrhu a pro realizaci vestavěných adaptivních systémů. Například obvod FPGA Virtex II Pro [33], který obsahuje 23616 slices, 49788 klopných obvodů, 852 vstupně-výstupních bloků, 232 vestavěných pamětí Block RAM (BRAM) a dva vestavěné procesory IBM PowerPC 405, byl využit pro akceleraci evolučního návrhu obrazových filtrů (kapitola 3.3). V FPGA je implementováno pole 8 x 4 programovatelných bloků, se kterým pracuje CGP. Nejedná se o evoluci na úrovni konfiguračních dat FPGA, ale na úrovni chromozomu CGP, který je uložen v uživatelem vytvořeném 384-bitovém registru. Dále je obvodově realizována fitness jednotka. Trénovací obrázky o velikosti 128 x 128 pixelů jsou uloženy v externích pamětech SRAM. Genetické operace jsou

prováděny procesorem PowerPC, který je vestavěn v FPGA. Výpočet je řízen obvodově realizovaným řadičem.

První verze akcelerátoru běžící na kmitočtu 100 MHz umožňuje urychlit evoluční návrh 44krát oproti softwarové implementaci v procesoru Celeron 2,4 GHz [31]. Za 1 sekundu dokáže akcelerátor ohodnotit až 6000 kandidátních filtrů. Jeden běh evoluce je dokončen v průměru do 10 s. Nejnovější verze akcelerátoru obsahuje čtyři paralelně pracující pole programovatelných bloků a čtyři jednotky pro výpočet fitness. Dosahuje tak urychlení až 170 oproti referenční implementaci běžící na procesoru Celeron 2,4 MHz [32].

4.3 Evoluce na úrovni konfigurace FPGA

Cílem experimentu A. Thompsona z roku 1996 bylo nalezení obvodu, který pracuje jako tónový diskriminátor v programovatelném hradlovém poli XC6216 [28]. Tento čip je jedním z předchůdců v současnosti nejpopulárnější rodiny Virtex. Čip XC6216 byl výrazně jednodušší než FPGA na obrázku 7, ale hlavně podporoval parciální dynamickou rekonfiguraci a návrháři byl znám význam každého konfiguračního bitu. Nicméně stejně jako u moderních FPGA výrobce předpokládal, že obvod bude používán výhradně pro konstrukci číslicových systémů konvenčním způsobem a neočekával, že by elementy obvodu pracovaly např. jako analogové komponenty.

Cílový obvod – tónový diskriminátor – má generovat log. 1, pokud je vstupní signál nastaven na frekvenci 10 kHz, a log. 0, pokud je vstupní signál nastaven na frekvenci 1 kHz. Evoluce byla prováděna přímo na úrovni konfiguračního řetězce a pro kandidátní obvody bylo vyhrazeno pole 10 x 10 logických bloků, což je velmi málo, pokud by měla být úloha řešena konvenčním způsobem. K zakódování řešení bylo potřeba 1800 bitů. Protože bylo na úrovni konfiguračního řetězce „vše povoleno“, mohly vzniknout netradiční asynchronní obvody a dokonce obvody, které je vhodnější označit za analogové než digitální. Zajímavé je, že programovatelné elementy XC6216 mají zpoždění v jednotkách nanosekund, kdežto požadované chování je na úrovni milisekund (nebylo povoleno připojit žádné další externí členy). Fitness funkce vyčísluje rozdíl mezi průměrným napětím na výstupu pro sérii vstupů s frekvencí 1 kHz a sérii vstupů s frekvencí 10 kHz.

Nalezené řešení je asynchronní, používá jen několik málo programovatelných elementů (konvenční řešení by bylo výrazně složitější), vykazuje netypické zpětné vazby a využívá vlastnosti konkrétního materiálu, což je naprosto nevídané na této platformě určené pro číslicové obvody. Do dnešní doby se nepodařilo přesně zjistit, jak a proč tento obvod funguje! Bylo zjištěno, že když se nalezená konfigurace nahraje do jiného FPGA (ovšem stejného typu), pak obvod nefunguje. Pokud se podle architektury FPGA a této konfigurace vytvoří model nalezeného obvodu pro simulátor, výsledky simulace nesouhlasí s naměřenými hodnotami.

Výsledek byl interpretován tak, že je EA schopen využít pro řešení problému i ty vlastnosti platformy (např. chemické a fyzikální vlastnosti konkrétního polovodiče) a prostředí (teplotu, elektromagnetické pole, …), které běžný návrhář neuvažuje. Vzniká problém portability (přenositelnosti): řešení pracuje jen na platformě, kde bylo vyevolvováno a jen při okolních podmínkách, které existovaly během evoluce.

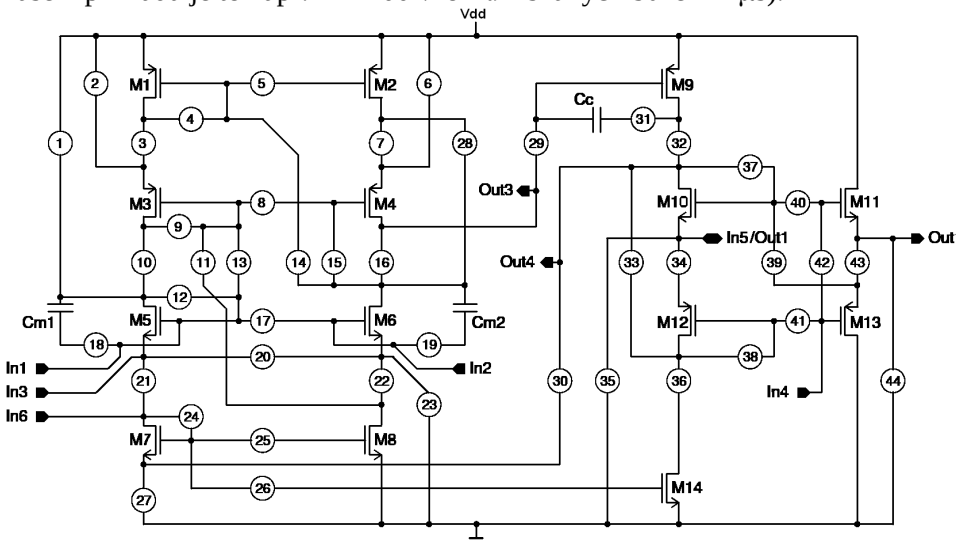
Thompson dále ukázal, že je možné evolučně v několika málo generacích „doladit“ řešení, které bylo nalezeno pro čip A, i pro čip B (stejného typu). Robustní řešení je také možné získat tak, že kandidátní obvody budeme během evoluce ověřovat v různých prostředích a na různých platformách.

Popsaný experiment demonstruje, že evoluční design umí něco, co člověk jako designér dobře nedokáže – precizně konfigurovat určité fyzikální systémy. Tento experiment byl motivací pro další výzkumníky.

4.4 Evoluce v extrémním prostředí

FPTA-2 je programovatelné pole tranzistorů, které obsahuje 64 programovatelných buněk. Buňky je možné programovat a propojovat pomocí vhodné konfigurace. Každá z buněk obsahuje 14 tranzistorů a 77 propojek (obrázek 9). Tento obvod byl použit výzkumným týmem z NASA JPL pro evoluční návrh analogových i digitálních obvodů na úrovni tranzistorů [26]. Zejména byl zkoumán vliv extrémního prostředí (teplota, radiace) na obvody vytvořené v FPTA-2 a schopnost evoluce opravit obvody poškozené tímto prostředím.

Chromozom opět přímo reprezentuje konfigurační řetězec FPTA-2. Jedna buňka je konfigurována pomocí 77 bitů, které jsou zaslány do FPTA v pěti 16b slovech. Ohodnocení kandidátního obvodu probíhá tak, že počítač generuje stimuly pro FPTA (např. při evoluci logického členu NOR na úrovni tranzistorů generuje každou ze čtyř vstupních kombinací po dobu 1 μs – viz obrázek 10), potom čte výstup FPTA, převede ho do digitální podoby a vypočítá fitness hodnotu, která je definována jako rozdíl mezi hodnotou změřenou na výstupu FPTA a požadovanou hodnotou (ideální odezva NOR vyznačená šedou barvou na obrázku 10) pro K vzorků získaných během evaluace (v našem příkladu je to např. K = 200 vzorků získaných během 4 μs).

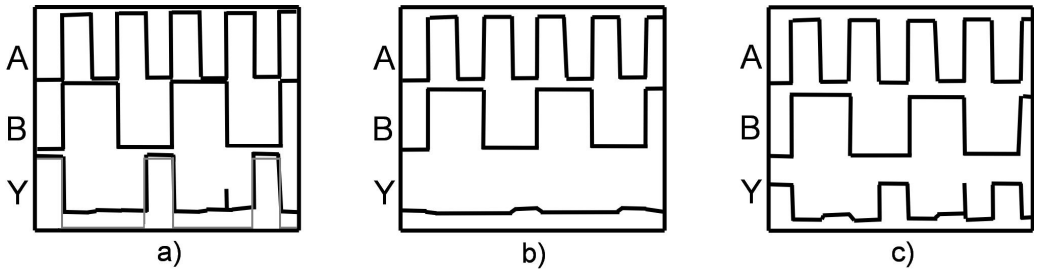


Obr. 9. Jedna programovatelná buňka čipu FPTA-2. Kroužek označuje programovatelný spínač.

Při evoluci na úrovni tranzistorů je mnohem jednodušší pozorovat vliv různých externích faktorů na výsledek evoluce než na úrovni logických členů. Častým jevem je např. situace, kdy evolučně navržený logický člen NAND pracuje korektně pouze při té frekvenci vstupních dat, která byla použita při výpočtu fitness hodnoty.

Za extrémní pro elektroniku považujeme teploty mimo rozsah -65ºC až 125ºC [4]. Při nízkých teplotách je nedostatek nosičů v polovodičích (protože dopanty nejsou dostatečně ionizovány). Při vysokých teplotách přestává pracovat PN přechod, nekontrolovaně přes něho protéká proud. Radioaktivní záření způsobuje u CMOS tranzistoru změnu závislost proudu I_d na napětí hradla, což může mít neodstranitelný vliv, pokud je obvod vystaven určité dávce. SEU (Single Event Upset) je naopak jev dočasný, kdy dochází k náhodnému překlopení bitu paměti. Nahráním původní informace do paměti dojde k opravě poruchy. Tradičně se funkce elektroniky v extrémním prostředí, které je typické pro vesmírné aplikace, zajišťuje zaváděním různých kompenzačních obvodů nebo použitím speciálních materiálů.

Experimentálně bylo prokázáno, že EA může obnovit funkci obvodu (např. logického hradla, DA převodníku apod.) vytvořeného a funkčního při pokojové teplotě, který je následně vystaven vlivu extrémního prostředí a přestává v něm pracovat. Toto extrémní prostředí způsobuje, že komponenty použité v obvodu se stávají komponentami s jinými elektrickými vlastnostmi. EA je využit, aby z těchto pozměněných komponent sestavil nové řešení. Jednalo se o znovuobnovení funkce pro nízké teploty (do –189 ºC), vysoké teploty (do 300 ºC) a radioaktivní záření (do kumulativního ozáření 250 krad) [25, 26, 34]. Příklad odezvy hradla NOR v extrémním prostředí a odezvy získané pro evolučně obnovené zapojení je znázorněn na obrázku 10.



Obr. 10. (a) Chování hradla NOR při pokojové teplotě. Šedá čára vyznačuje ideální požadovanou odezvu hradla. (b) Ztráta funkce při teplotě 326 ºC. (c) Obnovená funkce při teplotě 326 ºC. Vstupní signály jsou označeny A a B, výstup je označen Y.

4.5 Evoluce „in materio“

Při studiu experimentu A. Thompsona si J. Miller uvědomil, že EA může být velmi dobrým nástrojem pro automatickou „konfiguraci“ vhodného materiálu tak, aby prováděl požadovanou funkci – vznikl tak koncept evoluce „in materio“ [5].

Jeden z navržených scénářů vypadá takto: K vhodnému materiálu jsou na určitá místa připojeny elektrody, jejichž prostřednictvím je aplikováno elektrické konfigurační napětí. Při vhodně nastavených hodnotách konfiguračního napětí dochází ke změnám materiálu na molekulární úrovni. Byly testovány různé materiály. Pro experiment byl vybrán LCD displej. Je známo, že orientaci molekul kapalných krystalů je možné řídit elektrickým polem. Změna orientace těchto molekul potom mění optické a elektrické vlastnosti kapalných krystalů. Cílem evoluce je tedy najít taková konfigurační napětí, která způsobí takovou konfiguraci kapalných krystalů, že budou provádět požadovanou funkci. Podařilo se vyevolvovat různá chování, např. již zmíněný tónový diskriminátor, logické obvody nebo kontrolér pro robota [5].

Zde je nutné poznamenat, že není znám konvenční způsob, kterým by bylo možné takto materiál „nakonfigurovat“. Tento výsledek je významný z pohledu stále intenzivnějšího výzkumu v oblasti nanotechnologií, kdy se výzkumníci snaží různými technikami řídit vlastnosti a chování materiálů na úrovni atomů a molekul.

V oblasti molekulární elektroniky vznikla platforma nazvaná NanoCell, což je pole sebesetavujících se kovových částic propojených molekulárními přepínači [29]. Na určitá místa jsou připojeny elektrody, prostřednictvím kterých je možné aplikovat proudové pulsy a tím měnit konfiguraci NanoCell. Dochází ke změnám voltampérové charakteristiky molekuly nitroanilinu a tím i ke změně chování NanoCell. Pomocí EA byly na této platformě vytvořeny různé obvody, např. logická hradla.

4.6 Evoluce kvantových chování

Bartels a kol. se zabývali tvarováním laserových pulsů s cílem optimalizovat kvantové interference v dutém vlnovodu o průměru 175 μm, který byl naplněn argonem [1]. Tvarování bylo umožněno použitím deformovatelného zrcadla, jehož parametry byly hledány EA. Při tomto procesu vzniká řada harmonických. Některé měly být zvýrazněny, jiné potlačeny. Při experimentu bylo vygenerováno mnoho dat, která byla následně analyzována a porovnána s teoretickými predikcemi. Byla zjištěna chování, o kterých se doposud vědci domnívali, že nejsou fyzikálně vůbec možná (např. antikorelované harmonické v doméně attosekund). Podařilo se tak automatickým způsobem řídit kvantový systém, což před tím nebylo možné. Očekává se využití těchto principů v oblasti nanotechnologií.

5 Problém implementace a evoluční návrh

V předchozí kapitole jsme ukázali, že EA je schopen vytvářet elementární výpočetní systémy na konvenčních platformách (jako jsou rekonfigurovatelné číslicové a analogové obvody) i na speciálních, avšak jistým způsobem rekonfigurovatelných, platformách. Dále jsme ukázali, že EA může využít pro realizaci požadovaného chování specifické vlastnosti platformy i prostředí. Konvenční návrhové techniky prakticky nemají možnost využít tyto netradiční zdroje, které EA použít umí. Hlavním důvodem je, že tyto konvenční techniky vyžadují existenci abstraktního modelu

platformy. Evoluční návrh používající metodu „vygeneruj řešení a otestuj ho“ model platformy nepotřebuje.

Evolučně navržené výpočetní systémy představují velmi zvláštní a doposud prakticky nezkoumanou třídu výpočetních systémů. Pro další úvahy vyjděme z předpokladu, že EA je schopen v zadané fyzické platformě v rozumné době vyevolvovat takové chování, které vyžaduje specifikace zadaná uživatelem. V dnešní době nebude složitost takových systémů velká, ale můžeme očekávat pokrok v tomto směru. Je tak vytvořena implementace, která plní požadovanou funkci. Obecně ale nerozumíme, jak a proč řešení funguje. Intuitivně předpokládáme, že takový systém je výpočetním systémem (mechanismem). Z pohledu teorie implementace to však vůbec jasné není.

5.1 Problém implementace

V literatuře orientující se na přesah informatiky do filozofie je zkoumána otázka: Kdy je fyzikální systém výpočetním mechanismem? Jako stěžejní pro odpověď na tuto otázku považujeme řešení *problému implementace*: jaký je vztah mezi abstraktním výpočtem a jeho fyzickou realizací? Existuje zde celá řada přístupů. Shrňme si základní postoje podle [23].

Putnam říká, že musí existovat zobrazení mezi abstraktními výpočetními stavy a fyzikálními stavy, kterými systém během výpočtu prochází. Je kritizován (např. Scheutzem), protože podle této definice potom jakýkoliv fyzikální systém může být chápán jako implementace libovolného výpočtu. Scheutz preferuje začít ve fyzickém světě. Popisuje způsob, jakým je možné vstupy, výstupy a chování fyzikálního systému (který je popsán ve zvolené fyzikální teorii) vytvořit zobrazení na abstraktní výpočetní model. Tento přístup nevyžaduje koncept fyzického stavu [21]. Piccinini definuje výpočetní mechanismus jako mechanismus, jehož účelem je získat výstupní řetězec symbolů ze vstupního řetězce symbolů na základě obecně platného pravidla, které platí pro všechny vstupy a výstupy [19]. Definuje šest požadavků, které musí fyzikální systém splňovat, aby mohl být výpočetním mechanismem – zejména nezávislý pozorovatel musí být schopen „počítání“ identifikovat na základě studia systému. Potom je možné otázku, zda je fyzikální systém výpočetním mechanismem, formulovat jako hypotézu a ověřit její platnost na základě prozkoumání fyzikálního systému. Copeland, Johnson aj. vyžadují, aby existovala konzistentní interpretace symbolů uvnitř systému během výpočtu. Tato interpretace musí být deklarována před započetím výpočtu [11].

5.2 Třída evolučně navržených výpočetních zařízení

Jaké řešení má „problém implementace“ pro evolučně navržená výpočetní zařízení? Uvažme následující situaci: Cílem je najít konfiguraci rekonfigurovatelného zařízení (RZ), které bude realizovat funkci *F* požadovanou uživatelem. Abychom mohli definovat fitness funkci, musíme před započetím experimentu zvolit interpretaci fyzického chování RZ z pohledu vstupů a výstupů RZ. Můžeme např. přiřadit vhodným napěťovým úrovním na vstupech a výstupech logické hodnoty.

Evoluční algoritmus může objevit takové řešení problému (konfiguraci) *C*, které bude fungovat (tj. fyzicky realizovat požadované chování – funkci *F*) jen v daném rekonfigurovatelném zařízení (RZ). Konfigurace *C* nefunguje v simulátoru ani v jiném RZ stejného typu a funguje jen za podmínek (teplota, tlak, …), které existovaly na konci evoluce. V některých případech nejsme schopni vysvětlit, jak RZ realizuje funkci *F*. Inženýr prakticky není schopen konvenčním způsobem najít konfiguraci *C*. Jaké jsou tedy odpovědi na otázku: Je RZ konfigurované dle *C* a počítající *F* výpočetním mechanismem?

ANO: Pokud stačí, že pro zadané vstupy získáváme požadované výstupy dle interpretace, kterou jsme zavedli před tím, než byla spuštěna evoluce, potom je RZ výpočetním mechanismem. Pokud je RZ výpočetním mechanismem, potom funkce, které jsou vyčíslitelné pomocí RZ, jsou vyčíslitelné i na Turingově stroji.

NENÍ: Přísnější požadavek předpokládá, že symboly uvnitř RZ musí mít konzistentní interpretaci, která je zavedena před spuštěním evoluce. Tuto interpretaci však nelze zavést před započetím evoluce. EA může využít „cokoliv“ pro realizaci požadované funkce a my to dopředu principielně nemůžeme odhalit. Dokonce nelze zavést tuto interpretaci ani na konci evoluce po prozkoumání RZ! Tímto zkoumáním může být realizace poškozena a vytvořené chování ztraceno. Závažnějším argumentem však je, že EA může teoreticky pro zajištění a konstrukci řešení využít fyzikální jevy, které ještě nikdy nebyly popsány (viz kapitola 4.6). Tudíž zkoumáním RZ je nemožné pochopit činnost RZ, aniž bychom tyto jevy nejdříve poznali.

Rozhodnutí o tom, zda můžeme považovat evolučně navržené zařízení za výpočetní mechanismus, závisí na volbě definice výpočetního mechanismu. Pokud souhlasíme s tím, že nezávisí na tom, jak zařízení realizuje funkci, jsme schopni interpretovat jeho vstupně-výstupní chování jako výpočet a pokud jsem schopni tuto interpretaci zavést před započetím evoluce, potom můžeme zařízení považovat za výpočetní mechanismus. Pokud však je podstatné pro výpočetní mechanismy, že existuje korespondence mezi abstraktním a fyzickým, která je identifikovatelná pro nezávislého pozorovatele, potom naše zařízení není výpočetním mechanismem. Tabulka 1 porovnává tři typy fyzicky existujících systémů, jejichž chování bývá často interpretováno jako výpočet.

Tabulka 2. Vlastnosti výpočetních zařízení

Vlastnost	Běžné počítače	Mozek	Zařízení navržená EA
Reakce výstupů na zadané vstupy lze interpretovat jako výpočet	ANO	ANO	ANO
Abstraktní model výpočtu existuje před implementací	ANO	NE	NE
Uživatel může definovat požadované chování	ANO	NE	ANO
Inženýři mohou vyrobit	ANO	NE	ANO

Evoluční návrh obvodů a výpočetních systémů patří do oblasti elektroniky a počítačového inženýrství. V této analýze jsme se pokusili interpretovat evoluční návrh obvodů a výpočetních zařízení z pohledu teoretické informatiky. Ukázali jsme, že evolučně vytvořená výpočetní zařízení představují specifickou podtřídu výpočetních zařízení, která vykazuje zvláštní kombinaci vlastností, kterou nemají standardní počítače ani živé systémy, jejichž chování obvykle interpretujeme jako výpočet. Obecně není možné zjistit, jak a proč vyevolvované řešení funguje. Stávající teorie implementace, které řeší vztah mezi abstraktním a fyzickým počítáním, jsou potom nepoužitelné.

Klasická teoretická informatika definuje výpočet abstraktně, nejčastěji jako posloupnost přechodů mezi definovanými abstraktními stavy. Takto definovaný výpočet může být implementován pomocí různých fyzikálních principů. Vyevolvované výpočetní systémy jsou však závislé na své fyzikální podstatě a nemohou bez ní existovat. Pokud bychom odhalili abstraktní model, podle kterého provádí výpočet, potom bychom mohli tento výpočet přenést i na jiné platformy.

6 Patentová ochrana výsledků získaných evolučním návrhem

V kapitole 3.1 jsme se zmínili, že v soutěži Humies může získat ocenění evolučně navržené řešení, které je znovuobjevením patentovaného vynálezu, vylepšením patentovaného vynálezu nebo by dnes mohlo být kvalifikováno jako patentovatelný vynález. V této podkapitole se budeme zabývat otázkou, jak by v budoucnu mohl evoluční návrh ovlivnit chápání patentového práva. Nejdříve však popíšeme princip patentu a zejména problémy spojené s patentováním software.

6.1 O patentech obecně

Patent je výhradní právo, jenž přiznává stát vynálezu, který je nový, má tvůrčí úroveň a lze jej průmyslově využít [36]. Majitel patentu může legálně zabránit dalším osobám využívat vynález. Majiteli patentu je tak umožněno zajistit si příjem, který má kompenzovat náklady vynaložené na výzkumnou, vývojovou, realizační a další činnost spojenou se získáním patentu. Platnost patentu je obvykle 20 let a je omezena na určité teritorium. Majitel je ale povinen zveřejnit obsah vynálezu, který se tak stává veřejně známou součástí techniky. Tento princip je výhodný i pro společnost, protože podporuje vznik inovací. Ačkoliv nemůže patentovaný vynález nikdo kromě majitele legálně využívat, může se stát ihned po zveřejnění inspirací pro další vynálezce a vést k novým inovacím.

Aby byl vynález patentovatelný, musí splňovat několik požadavků. Především se musí jednat o *patentovatelný předmět*. Většinou je výslovně uvedeno, co patentovat nelze. V některých zemích například nelze patentovat software, který může být chráněn jiným nástrojem – autorskými právy. Dále přihlašovatel prokazuje *novost vynálezu* na základě podrobné rešerše stavu techniky. Vynález musí vykazovat tvůrčí úroveň, tj. musí být *nezřejmý* odborníkovi v příslušném technickém oboru (s přihlédnutím ke stavu techniky). Tento požadavek má zaručit, že patent nebude

udělen za pouhé vylepšení již existujícího řešení, ke kterému může dospět osoba s obvyklými schopnostmi a znalostmi v daném oboru. Dalším požadavkem je *průmyslová využitelnost* neboli užitečnost. Nelze totiž získat patent například za objevení nějakého teoretického fenoménu, u kterého není jasná využitelnost. A konečně, vynález musí být v přihlášce objasněn tak, aby jej mohl realizovat odborník v konkrétním technickém oboru.

V některých zemích (včetně České republiky) je možné získat tzv. *užitný vzor*, což je jednodušší než získat patent. Pro užitné vzory je typické, že požadavek na tvůrčí úroveň je nižší a řízení o udělení je kratší. Užitný vzor s příslušnou právní ochranou tak mohou být uděleny relativně rychle. Doba platnosti užitného vzoru je rovněž kratší (např. 4 roky). Současně s přihláškou užitného vzoru je obvykle podávána i řádná patentová přihláška.

6.2 Softwarové patenty

Tradičně jsou patentovány produkty a výrobní procesy. Patentová ochrana software existuje pouze krátkou dobu a jen v určitých zemích. V Evropě platí Evropská patentová úmluva, která explicitně vyjímá matematické postupy a počítačové programy ze seznamu patentovatelných předmětů. Nicméně je fakticky možné patentovat postupy, jejichž význačnou součástí je počítačový program. Naopak v USA se stalo velmi populární patentovat software. Přibližně 15% ze všech patentů je uděleno na software a počet žádostí o udělení softwarového patentu vzrostl za posledních 20 let 18krát [20].

Tento stav vede k situaci, kdy většina klíčových (částí) programů je patentována a není téměř možné vytvořit nový program, aniž by nebyl některý z existujících patentů porušen. Protože patenty na software donedávna neexistovaly, bylo možné po jejich zavedení získat celkem jednoduše patentovou ochranu na téměř jakýkoliv nový software, a přitom neporušit jiný patent. Toho využily velké společnosti, které mají prostředky na podávání a udržování širokého spektra patentů. Navíc je často velmi obtížné zjistit, zda konkrétní programová konstrukce je či není chráněna patentem, protože patentová přihláška popisující určitých vynález může obsahovat celou řadu patentových nároků, které s vynálezem souvisejí jen velmi málo. Dochází tak k blokování nových inovací, protože majitelé patentů namísto tvorby nových aplikací sledují, zda někdo neporušuje jejich patenty, a pokud se tak děje, požadují licenční poplatky [15]. Mezi velkými firmami se používá (prakticky bezplatné) křížové licencování, které dovoluje firmě A využívat patentované technologie firmy B a naopak. Malé firmy vlastnící jeden nebo několik málo patentů jsou v nevýhodě, protože za křížovou licenci sice získají možnost využívat část technologie patentovanou velkou společností, ale velká společnost získá bezplatně přístup k jejich výsledkům tvůrčího procesu.

6.3 Bude nutná nová interpretace patentového práva?

Pokud nahlédneme do patentových databází, tak zjistíme, že v souvislosti s evolučním hardware najdeme patentované:

- obvody navržené evolučně, např.:
  - US Patent 6847851: Apparatus for improved general-purpose PID and non-PID controllers (Koza et al., 2005)
  - užitný vzor UV 020017/2009: Nelineární obrazový filtr (Sekanina, Vašíček, 2009)
- evoluční algoritmy, např.:
  - US Patent 7117186: Method and apparatus for automatic synthesis of controllers (Koza et al., 2006)
  - US Patent 6526556: Evolutionary technique for automated synthesis of electronic circuits (Stoica et al., 2003)
- platformy pro evoluční návrh obvodů, např.:
  - US Patent 6728666: Evolvable circuit with transistor-level reconfigurability (Stoica et al., 2004)
  - US Patent 6222381: Self-configurable parallel processing system made from self-dual code/data processing cells utilizing a non-shifting memory (Durbeck a Macias, 2001)

V českém prostředí je možné získat patent na evolučně vytvořené řešení, popř. na rekonfigurovatelnou platformu použitou pro evoluční návrh. Není možné získat patent na nový evoluční algoritmus. V zásadě platí, že pokud evoluční algoritmus nalezne řešení problému, které splňuje požadavek na udělení patentu, je možné sepsat patentovou přihlášku popisující toto řešení a patent by měl být udělen. Není nutné uvést, že řešení bylo získáno pomocí evolučního návrhu.

V monografii [20], kterou sepsal přední americký právník zabývající se softwarovými patenty – Robert Plotkin, se uvádí, že tento stav je dlouhodobě neudržitelný a nevýhodný jak pro vynálezce tak i pro společnost. Plotkin analyzuje situaci, která by mohla nastat v blízké budoucnosti, kdy budou volně k dispozici systémy pro evoluční návrh použitelné pro rutinní generování vynálezů. Pro ty, kteří budou disponovat finančními prostředky na poplatky za patentové přihlášky a udržování patentů, nebude příliš obtížné vygenerovat velké množství vynálezů a získat příslušné patenty. Mohlo by dojít ještě k horší situaci než je v současné době v oblasti patentování software. Tato situace se nazývá *patentová záplava* (patent flood) a vede k zablokování nových inovací, protože klíčové vynálezy jsou v rukou několika mazaných vynálezců (savvy inventors). Současně je zpomalena činnost patentového úřadu, protože posuzování patentových přihlášek je prováděno lidskou silou a při jejich velkém počtu nebudou patenty uděleny v rozumném čase. Navíc je pro člověka často velmi obtížné pochopit, jak funguje evolučně vytvořený vynález. Dalším problémem je, že evoluční algoritmus v jednotlivých bězích často vygeneruje různá a přitom stejně kvalitní řešení. Evolučně vytvořený patentovaný vynález by někdo mohl obejít tak, že použije evoluční algoritmus a vytvoří alternativní řešení, které nebude předmětem patentové ochrany.

Plotkin nabízí řešení popsané situace, které je založeno na nové interpretaci stávajícího přístupu k udělování patentů. Zaměřme se na některé zajímavé postřehy související s důsledným aplikováním stávajících požadavků na přihlašované patenty.

Jedním ze základních požadavků je tvůrčí úroveň, tj. vynález musí být nezřejmý odborníkovi, který má „běžné dovednosti v příslušném technickém oboru“. Pokud by totiž posuzovatel patentu, který má rovněž „běžné dovednosti v příslušném technickém oboru“, měl k dispozici software pro evoluční návrh (ten by byl např. volně dostupný na internetu), dokázal by s jeho pomocí získat stejný nebo velmi podobný vynález. Tímto způsobem se ale řešení původně nezřejmé odborníkovi, který má „běžné dovednosti v příslušném technickém oboru“, stává zřejmým a dosažitelným pro odborníka, který má „běžné dovednosti v příslušném technickém oboru“ a tudíž není splněna jedna ze základních podmínek pro udělení patentu a žádost je třeba zamítnout.

Plotkin navrhuje, aby se v budoucnu patenty namísto evolučně vytvořeným řešením udělovaly *specifikacím*, tj. postupům, jak sestavit fitness funkci. Pokud je návrhový software s evolučním algoritmem volně dostupný, tak je to právě šikovně formulované zadání (specifikace), které vede ke vzniku nových a patentovatelných řešení. Z patentové přihlášky by ale muselo být zřejmé, že uvedená specifikace vede k patentovatelným vynálezům. Pokud by byla patentována specifikace, znamenalo by to současně ochranu i pro všechny výsledky evolučního návrhu vytvořené podle této specifikace. Patent by se potom vztahoval na celou třídu vynálezů, ne pouze na jeden, jako je tomu doposud. Zdá se, že zavést patentovou ochranu jen pro specifikace je užitečnější než pro celé evoluční návrhové systémy. Volná dostupnost návrhových systémů (nejen v oblasti evolučních algoritmů) se totiž ukázala jako velmi důležitá pro rozvoj inovací.

Doposud platilo, že patentové právo určuje, *kdo vlastní vynález*. Pokud by bylo možné patentovat specifikace, potom by patentové právo vlastně určovalo, *kdo má právo vytvářet vynálezy*. A v tom je velký rozdíl.

7 Závěr

V této kapitole byly objasněny základní principy evolučního návrhu hardware. Ukázali jsme, že existuje podstatný rozdíl mezi evolučním návrhem prováděným pomocí simulátoru a evolučním návrhem probíhajícím přímo v rekonfigurovatelném zařízení. V současnosti existují stovky evolučně navržených řešení, jenž můžeme považovat za konkurenceschopné těm řešením, které byly vytvořeny inženýry-návrháři konvenčním způsobem. Při analýze problému implementace jsme ukázali, že evolučně navržené systémy představují specifickou podtřídu výpočetních zařízení, která vykazuje zvláštní kombinaci vlastností, kterou nemají standardní počítače. Z praktického pohledu vzniká problém s patentováním evolučně vytvořených vynálezů, protože stávající interpretace patentového zákona není vyhovující. Vzhledem k tomu, že je evoluční návrh stále více populární, je možné předpokládat, že zasáhne do profesního života řady dalších lidí.

Poděkování: Tato kapitola vznikla za podpory projektů GAČR P103/10/1517 *Natural computing na nekonvenčních platformách* a MŠMT 21630528 *Výzkum informačních technologií z hlediska bezpečnosti*.

Literatura

[1] Bartels, R. et al.: Learning from Learning Algorithms: Applications to attosecond dynamics of high-harmonic generation. *Physical Review A*. **70**(1):1-5 (2004)

[2] Bentley, P. (ed): *Evolutionary Design by Computers*. Morgan Kaufmann Publishers, San Francisco CA 1999

[3] Drechsler, R.: *Evolutionary Algorithms for VLSI CAD*. Boston: Kluwer Academic Publishers, 1998.

[4] Extreme Temperature Electronics - Tutorial (2005) http://www.extremetemperatureelectronics.com

[5] Harding, S., Miller, J., Rietman, E.: Evolution in materio: Exploiting the physics of materials for computation. *International Journal of Unconventional Computing*. **4**(2):155–194 (2008)

[6] Higuchi, T. et al.: Evolving Hardware with Genetic Learning: A First Step Towards Building a Darwin Machine. In: *Proc. of the 2nd International Conference on Simulated Adaptive Behaviour*. MIT Press, Cambridge MA 1993, s. 417-424

[7] Higuchi, T., Liu, Y., Yao, X.: *Evolvable Hardware*. Springer Verlag, 2006

[8] Human Competitive Awards in Genetic and Evolutionary Compuation – Humies 2004–2010. http://www.genetic-programming.org/hc2005/main.html

[9] Hwang, H., Haddad, R.: Adaptive median filters: new algorithms and results. *IEEE Transactions on Image Processing*. **4**(4):499–502 (1995)

[10] Chan, R.H., Ho, C.W., Nikolova, M.: Salt-and-pepper noise removal by median-type noise detectors and edge-preserving regularization. *IEEE Transactions on Image Processing*. **14** 1479–1485 (2005)

[11] Johnson, C. G.: What Kinds of Natural Processes Can be Regarded as Computations? In: *Computation in Cells and Tissues: Perspectives and Tools of Thought*. Springer, Berlin 2004, s. 327–336

[12] Koza et al.: *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*, Springer, 2005

[13] Kvasnička, V., Pospíchal J., Tiňo P.: *Evolučné algoritmy*. Vydavatelství STU Bratislava, 2000

[14] Lohn, J. D., Hornby, G. S.: Evolvable hardware: Using evolutionary computation to design and optimize hardware systems. *IEEE Computational Intelligence Magazine* **1**(1): 19–27 (2006)

[15] Matyska, L.: Softwarové patenty. *Zpravodaj ÚVT MU*. **15**(4): 16-20 (2005)

[16] Miller, J., Job, D., Vassilev, V.: Principles in the Evolutionary Design of Digital Circuits - Part I. *Genetic Programming and Evolvable Machines*. **1**(1): 8-35 (2000)

[17] Novák, O. et al.: *Handbook of Electronic Testing*. Vydavatelství ČVUT Praha, 2005

[18] Pečenka, T., Sekanina, L., Kotásek, Z.: Evolution of Synthetic RTL Benchmark Circuits with Predefined Testability. *ACM Transactions on Design Automation of Electronic Systems.* **13**(3):1–21 (2008)

[19] Piccinini, G.: Computations and Computers in the Sciences of Mind and Brain. PhD thesis, University of Pittsburgh, 2003, 323 s.

[20] Plotkin, R.: *The Genie in the Machine*. Stanford Law Books, 2009

[21] Scheutz, M.: When Physical Systems Realize Functions … *Minds and Machines*. **9**(2): 161-196 (1999)

[22] Sekanina, L.: Evolvable Components: From Theory to Hardware Implementations. Springer Verlag, Berlin 2004

[23] Sekanina, L.: Evolved Computing Devices and the Implementation Problem. Minds and Machines **17**(3): 311-329 (2007)

[24] Sekanina, L. et al. Evoluční hardware. Od automatického generování patentovatelných invencí k sebemodifikujícím se strojům. Academia Praha, 2009

[25] Stoica, A., Zebulum, R., Keymeulen, D., Ferguson, I., Duong, V., Guo, X.: Evolvable hardware techniques for on-chip automated reconfiguration of programmable devices. *Soft Computing - Spec. Issue on Evolvable Hardware*. **8**(5): 354-65 (2004)

[26] Stoica, A., Keymeulen, D., Arslan, T., Duong, V., Zebulum, R., Ferguson I., Guo, X.: Circuit Self-Recovery Experiments in Extreme Environments. In: *Proc. of the 2004 NASA/DoD Workshop on Evolvable Hardware*, IEEE Computer Society, Los Alamitos 2004, s. 142-145

[27] Šonka, M., Hlaváč, V.: Image Processing, Analysis, and Machine Vision. CL-Engineering, 3 vyd., 2007

[28] Thompson, A.: *Hardware Evolution: Automatic Design of Electronic Circuits in Reconfigurable Hardware by Articial Evolution*. Distinguished Dissertation Series, Springer, London 1998

[29] Tour, J. M.: *Molecular Electronics*. World Scientic 2003

[30] Vassilev, V., Job, D., Miller, J.: Towards the Automatic Design of More Efficient Digital Circuits. In Proc. of the 2nd NASA/DoD Workshop on Evolvable Hardware, IEEE Computer Society, 2000, s. 151-160

[31] Vašíček, Z., Sekanina, L.: An Evolvable Hardware System in Xilinx Virtex II ProFPGA. *International Journal of Innovative Computing and Applications.* **1**(1): 63-73 (2007)

[32] Vašíček, Z., Sekanina, L.: Efficient Hardware Accelerator for Symbolic Regression Problems, In: 5th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, Znojmo, CZ, MUNI, 2009, s. 192-199

[33] Xilinx: Virtex-II Pro Platform FPGAs: Complete Data Sheet (2005) URL: http://www.xilinx.com/partinfo/ds031.pdf.

[34] Zebulum, R., Stoica, A., Keymeulen, D., Sekanina, L.: Evolvable Hardware System at Extreme Low Temperatures. *LNCS* 3637, Springer, 2005, s. 37-45

[35] Zelinka et al.: *Evoluční výpočetní techniky. Principy a aplikace*. Technická literatura - BEN, 2009

[36] Vynálezy pro budoucnost – Úvod do problematiky patentů pro malé a střední podniky. WIPO 2006 http://www.wipo.int/ebookshop